EIC Proposal for a Scalable, Deadtime-free, Trigger-less Readout Scheme

R. Abruzzio, J. Balewski, J.C. Bernauer, B. Buck, R. Corliss, C. Epstein, D.K. Hasell, R. Milner, R. Redwine, J. Stevens, and M. Williams
Laboratory for Nuclear Science
Massachusetts Institute of Technology
Cambridge, MA 02139

June 28, 2014

Abstract

We propose to develop a modular, scalable, streaming readout scheme for future EIC experiments. To do this we will produce a sample system to study the design and implementation. The resulting scheme would permit a cost-effective, deadtime-free readout system and a flexible software system replacing the traditional hardware trigger. Our approach differs from those being adopted by other experiments like LHCb and ALICE. We will organise the data on FPGAs and complete the event build by sending them specific computer nodes via Ethernet. Event classification algorithms will run on virtual machines on these same computer nodes. We will explore the boundaries of offloading peak processing needs to local and commercial cloud computing providers to lay the ground work toward integration of the future EIC DAQ with the offline computing structure of a future EIC host laboratory.

With the high luminosity expected at EIC such a streaming readout system will be necessary. It is important to make this study and decide of a scheme now, so the detector systems currently being developed can be optimized for chosen readout system.

1 Introduction

Until very recently nuclear and particle physics experiments collected data by connecting the detector signals to commercially available modules (*e.g.*, ADCs, TDCs) or to specially designed electronics (*e.g.*, ASICs) that converted the signals into a digital format suitable for computer analysis. Typically these units stored or buffered the data until a readout request was received, after which the data would be serially transmitted to the data acquisition computer(s) or pipelined pending a subsequent readout request or abort signal. The readout request would be sent to some or all of these units depending on the result of a trigger algorithm that was fed a small volume of data that could be made available quickly. The readout decision had to be made timely enough that the various buffers would not overwrite the event before it was requested. Once requested, the readout time was typically large compared to the raw rates in the detectors, imposing a finite deadtime before another event could be successfully read out. This scheme imposes limits on the complexity of the trigger logic and on the maximum rate of accepted events that become more prohibitive as the instantaneous luminosity of an experiment increases.

In order to take full advantage of their high luminosities, many modern experiments[1–3] take a different approach to data acquisition. Instead of sending readout requests for individual events, the entirety of the data produced by the detectors is processed and streamed in real-time. The various data streams are collected together in arrays of networked computers which combine the various streams to form candidate events. The data for these candidate events are then passed on to a further array of computers for analysis and storage. For example, the LHCb scheme is shown in Fig. 1.

The future Electron Ion Collider,[4] (EIC) with an expected luminosity on the order of 10^{34} cm^{-2}s^{-1}, will need to have a readout system with a similar capability. This scheme needs to be developed now so that the EIC detectors being designed do so with the adopted readout scheme in mind. For example the detectors should not require a trigger. Similarly the detector, with knowledge of the method used to combine hits to form events and how reconstruction will be performed, can optimise its readout accordingly.

We propose to develop and study a design for a streaming, high rate, scalable readout system. We focus on the data transport from the front-end electronics (FEE) to the analysis computers but also address the event classification problem. While a single detector channel stream has the information for all times, the event analysis needs the information from all channels for the time period of a potential event. Our approach collects the data accordingly and transmits it to separate CPUs.

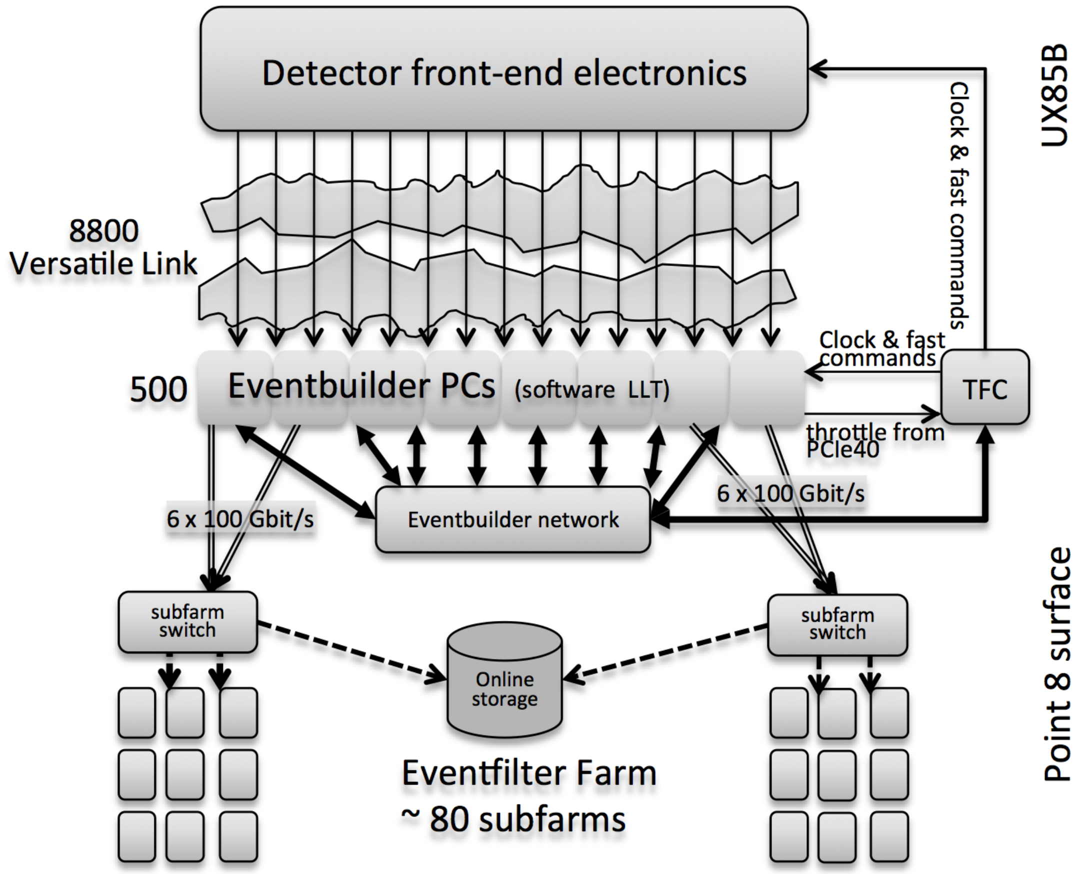

Figure 1: The LHCb readout scheme[1] uses 8,800 optical fibres to carry the digitized, zero-suppressed data from the detector 300 m to the surface based data acquisition system. The streams of data are collected in an array of 500 networked computers that build the events. The events are then passed through high speed switches to sub-farms of computers for analysis and storage.

While we do not design any front end electronics specifically for this proposal (we will simulate the data from the detector using a FPGA based data generator), we presume that the front end electronics for the detector will provide streamed data for each channel without the need for an external trigger. As will be described in Sec. 2, the front-end electronics will stream zero-suppressed data and add a time stamp and channel identifier to each hit. This simple step done in the front end will greatly simplify the sorting of the various hits in the readout network.

Many data channels will be combined to efficiently use the available bandwidth of the data distribution modules (DDM). These modules will use the time stamp information to collect data within a certain time window and transfer that information as a package to a specific analysis computer for reconstruction and storage.

These data distribution modules provide the functionality of the event builder PCs of the LHCb and ALICE proposals. We believe the proposed architecture using FPGAs will be more efficient, simpler, and more cost effective.

2 Design Approach

An overview of the proposed readout scheme is illustrated in Fig. 2. The readout is implemented in two stages. The first stage (shown in yellow) is a hierarchical readout network which combines the information from many channels. This concatenation of the input streams optimally uses the available bandwidth on the distribution modules. The second stage (shown in blue) consists of the distribution network that transfers the information from these modules to a specific CPU according to the time-slice for classification and conditional storage.

The data acquisition system is composed of several components:

- Front end electronics (FEEs): This includes the analog front end, the analog-to-digital converter (ADC) (or time stamping TDC, *etc.*), and some logic to produce zero-suppressed information, which is output via a high speed fiber link. In this proposal, this component will be simulated by an FPGA based data generator.

- Concatenation modules (CMs): These boards have several fiber links as input which are combined to form a single output.

- Data distribution modules (DDMs): These modules receive as input the data from the FEE or CM boards and send the data corresponding to the same time slice to the same CPU node via a switched network, such as Ethernet.

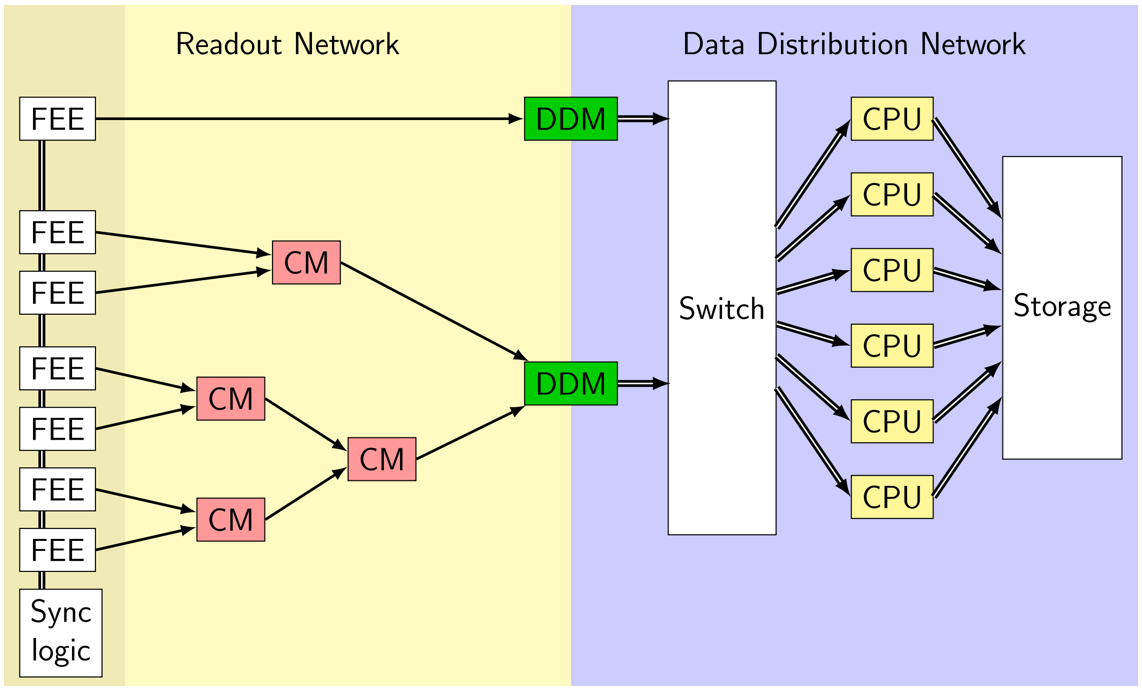

Figure 2: The proposed readout scheme, divided into readout (left, yellow background) and distribution (right, blue background) networks. Detector information is digitized by the front-end electronics (FEEs). Depending on the number of channels and the data rate within an FEE module, the data stream can either be directed directly (top) to a data distribution module (DDM), or concentrated (bottom) using concatenation modules (CMs), which can be further cascaded up to the bandwidth limits of the connections. The DDMs sit at the border between the readout and distribution networks, sorting the incoming data packets by time-slice and sending the data as Ethernet packets into a standard commercial Ethernet switch with appropriate bandwidth. The DDMs label all data packages from the same time-slice with the same target address so they reach the same CPU. Each CPU receives the full information for a time-slice and can analyze the event. If the event is interesting, it will be written out to the storage backend. Synchronization of the FEE clocks, which is necessary to properly label time-slices, is maintained by a distributed clock signal. Only the modules on the left (darker yellow background), *i.e.*, FEE and clock sync logic, are detector specific.

- Event reconstruction is performed on the CPU for each time slice.
- Accepted events are committed to long time storage.

Except for the header of a data packet, which must contain a time stamp, the readout system is independent of the type of data sent by the FEE.

We introduce the concept of the time-slice, a period slightly longer than the longest event duration, *i.e.*, the longest possible time interval between two signals in the detector that are associated with the same physics event. Time-slices are non-overlapping and labeled sequentially, hence any event is contained within at most two consecutive time-slices.

If different detectors have widely varying intrinsic event durations, it is feasible to have different time-slice durations for different parts of the readout network. The only requirement is that all sources feeding into the same DDM have synchronized time-slice numbers. For example, in a two detector setup consisting of a fast and slow detector, the time-slices of the fast detector can be collected and analyzed before the slow detector data assembly is finished. Then, in a second step, only already pre-analyzed and marked-as-relevant data from the fast detector have to be matched to the information of the slow detector by the CPU farm.

The readout network collects packets tagged with each time-slice number. The DDM looks up the target network address using the time-slice number. For each target CPU, the DDM manages a FIFO buffer. When the FIFO is close to full, the aggregated data is sent over the Ethernet network to the target CPU. The mapping of time-slice to CPU is flexible, and some data must be replicated to ensure deadtime free operation. In the most extreme case, each time-slice is mapped to two CPUs, once together with the preceding time-slice, and once with the next time-slice. In this way, each event is fully contained in the data received by at least one CPU. This mode has the smallest latency, but also the highest bandwidth demand, as each datum is sent twice. Fig. 3 illustrates the data flow for this case.

It is also possible to map blocks of consecutive time-slices to the same CPU. This can dramatically reduce the overhead to guarantee that any event is completely in the buffer on one CPU since only the data for the last time-slice of each block has to be duplicated. For example, sending 1000 consecutive time-slices to a given CPU, the over head is reduced to 1 per mil. In this mode of operation, forgoing the duplication would create negligible deadtime. One has a minimal bandwidth overhead, but needs larger buffers in the switching fabric and introduces additional latency.

The proposed design is flexible in this regard and allows experimenters to choose an optimal balance of bandwidth and buffer size for their needs.

The CPUs now have the full information for a time period covering a possible interesting event (or events). They can now decide whether to keep the data or not. If this decision can be computed before the next time-slice block arrives, it can be made on that machine directly. If not, the target CPUs can act as gateways to a larger cluster. Since they already have the complete information to an event, they simply have to pass it on to the next available CPU in the cluster.

3 R&D

3.1 Front-End Protocol

The front-end electronics will be detector and experiment specific and can only be developed together with the detectors. We therefore focus on developing a protocol for transmitting data through the network and producing a reusable set of firmware modules that will allow the detector and experiment designers to easily integrate their front end electronics with the data acquisition system. The full design of the front-end electronics is a task for the group developing the specific detector keeping in mind the requirements of the readout system..

A hypothetical design for a front-end board is shown in Fig. 4. In this design the analog data are sampled and then run through a zero-suppression algorithm producing information about, *e.g.*, hit amplitude and time-of-event arrival.

As part of our initial investigations, we will design data generators to simulate the front-end electronics of a detector and feed data into the rest of the data acquisition system. These will run on FPGA development boards and will be available for testing throughout the development process. The simulated design has a variable number of input channels. We assume that each channel is being sampled at 40 MHz with 8 bit resolution. We expect per channel event rates of 100 kHz. After zero-suppression, each event is described by a payload of 10 bytes, yielding an average data rate of 1 MBytes/s per channel. A 256 channel module will produce an average rate of 2 GBits/sec which is easily manageable by an FPGA driving a commercially available optical link.

These specifications (channels per board, frequency, resolution and event rate) represent what we expect some typical detector technologies may utilize. However, we plan to produce a robust protocol and firmware core which can be used with a broad range of detector specifications.

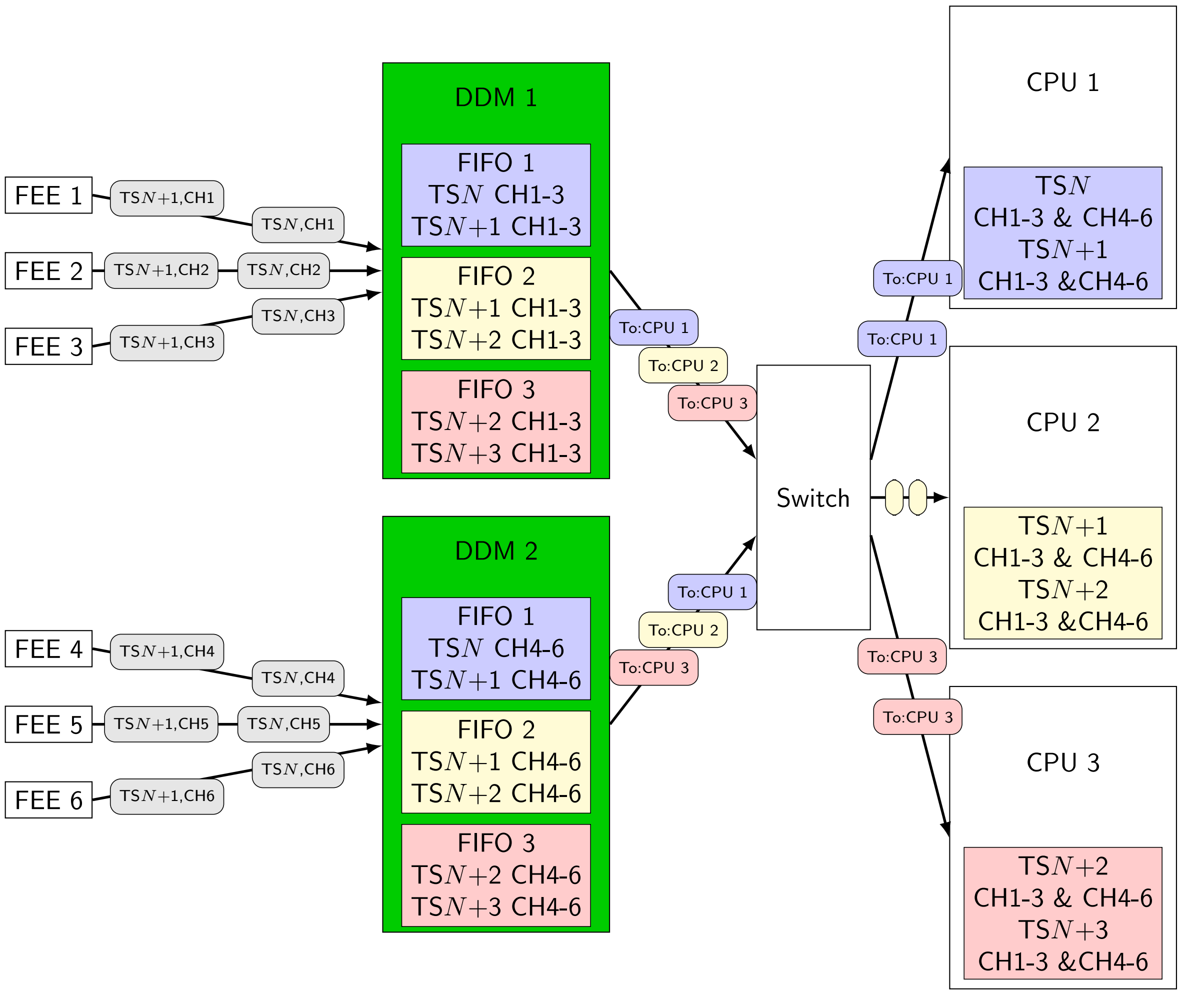
DDM 1
FIFO 1 TSN CH1-3 TSN+1 CH1-3
FIFO 2 TSN+1 CH1-3 TSN+2 CH1-3
FIFO 3 TSN+2 CH1-3 TSN+3 CH1-3
DDM 2
FIFO 1 TSN CH4-6 TSN+1 CH4-6
FIFO 2 TSN+1 CH4-6 TSN+2 CH4-6
FIFO 3 TSN+2 CH4-6 TSN+3 CH4-6
FEE 1
FEE 2
FEE 3
FEE 4
FEE 5
FEE 6
TSN+1,CH1
TSN,CH1
TSN+1,CH2
TSN,CH2
TSN,CH3
TSN+1,CH3
TSN+1,CH4
TSN,CH4
TSN+1,CH5
TSN,CH5
TSN,CH6
TSN+1,CH6
To:CPU 1
To:CPU 2
To:CPU 3
Switch
CPU 1
TSN CH1-3 & CH4-6 TSN+1 CH1-3 &CH4-6
CPU 2
TSN+1 CH1-3 & CH4-6 TSN+2 CH1-3 &CH4-6
CPU 3
TSN+2 CH1-3 & CH4-6 TSN+3 CH1-3 &CH4-6

Figure 3: Data flow of the readout for a small scale system. Six FEE modules, each serving one channel in this example, send their data stream to two DDMs. Each link transports the information for all time-slices (TS), but only for the single channel (CH). The DDMs sort the incoming stream into FIFOs by time-slice number, duplicating the information into two FIFOs each. The data of a full FIFO are sent via Ethernet to the switch, addressed to the corresponding CPU. The CPU receives data from all DDMs for each time-slice pair assigned to it. Once the full information of an event is available, the algorithms running on the CPUs can decide whether to keep or drop the data.

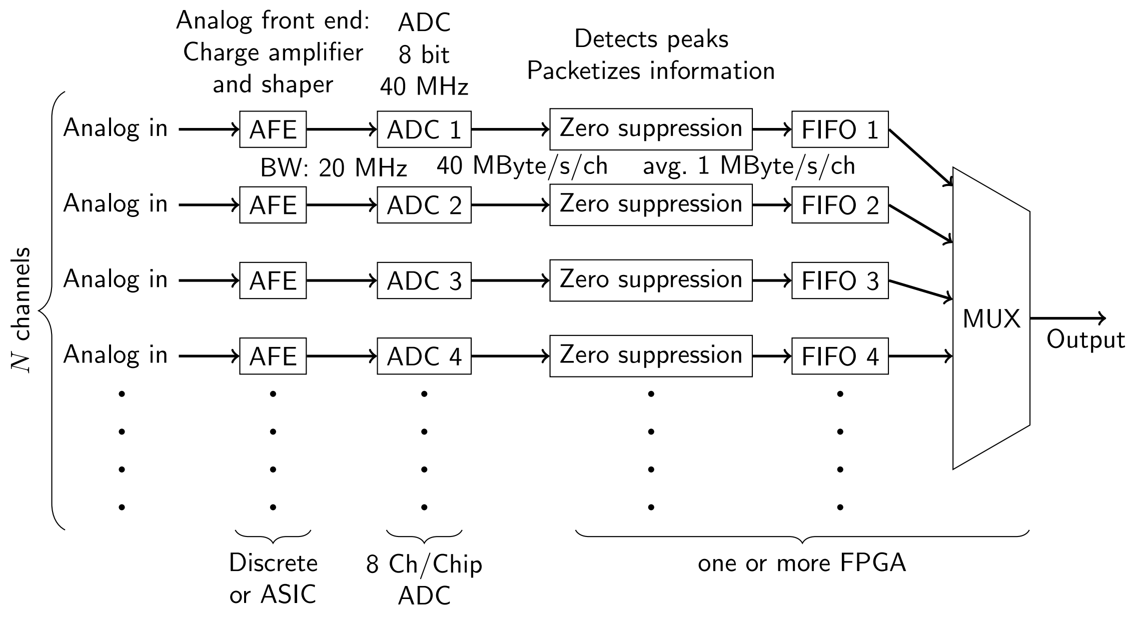

Figure 4: A hypothetical front end electronics (FEE) board has N parallel input stages. First, the analog input is processed by the analog front-end (AFE), which can be realized either with discrete components or highly integrated ASICs. This part could be separated out into a separate module, close to the actual detector. Behind the AFE, the signal bandwidth is limited to about 20 MHz, which reduces the requirements for the cable connection between AFE and ADC. The ADC stage can be realized with commercial 40 Megasamples/s, 8 channel ADCs. It digitizes the analog signal and produce an output stream of 40 MByte/s. The zero-suppression units detect peaks in this digital stream and generate packetized information about peak timing and shape. The data is tagged with the time-slice number at this stage (not shown) and moved into a FIFO buffer. The output mux mixes the packetized information onto the output data stream.

3.2 Concatenation Modules

Different front-end boards will have different maximum data output rates, depending on the number of channels, payload size and event rate. To utilize the DDM boards fully, we are planning to build concatenation modules (CMs), which combine multiple input streams to a single output stream. In R&D, we have to decide whether these boards should be "dumb", just multiplexing the input streams, or "intelligent", by combining input packages with the same time stamp. While the latter is more logic intensive, it also decreases the protocol overhead on the output link.

Initially, CMs will be produced using FPGA development modules attached to commercially available optical transceiver modules. This will let us test various aspects of the design before committing to a reference design. The reference design for this board, along with the firmware to run it, will make up part of the deliverables. Having a working reference design will allow groups with limited engineering access to integrate with the data acquisition system with minimal engineering.

3.3 Data Distribution Modules

The data distribution modules (DDMs) are the core element of the design. They receive a data stream from CM or FEE boards which contains time-slice tagged and packetized data. These modules sort the data by time-slice and send data to an appropriate computer via Ethernet using commercially available optical transceivers.

Significant effort is required to define the inner workings of this protocol and to make it as universally useable as possible. We will initially develop this protocol and firmware to run on an FPGA development board. This will be connected to a high-speed Ethernet switch and a small computer cluster in order to verify the performance of our design.

Once a protocol and design has been proven, we will develop a reference DDM design and firmware to run the DDM hardware. It is highly likely that an experiment or detector using our data acquisition system would be able to use our DDM design with no modifications.

3.4 Event-Processing Nodes

The more detailed block diagram of the compute cluster is shown in Fig. 5. Once a complete event arrives from multiple DDMs to a single compute node (see Fig. 3), the event classifier algorithm is executed on one or more of the cores of this node. In general, there may be multiple stages of the classifier algorithm with increasing complexity, but reduced number of input events to reduce the average decision time.

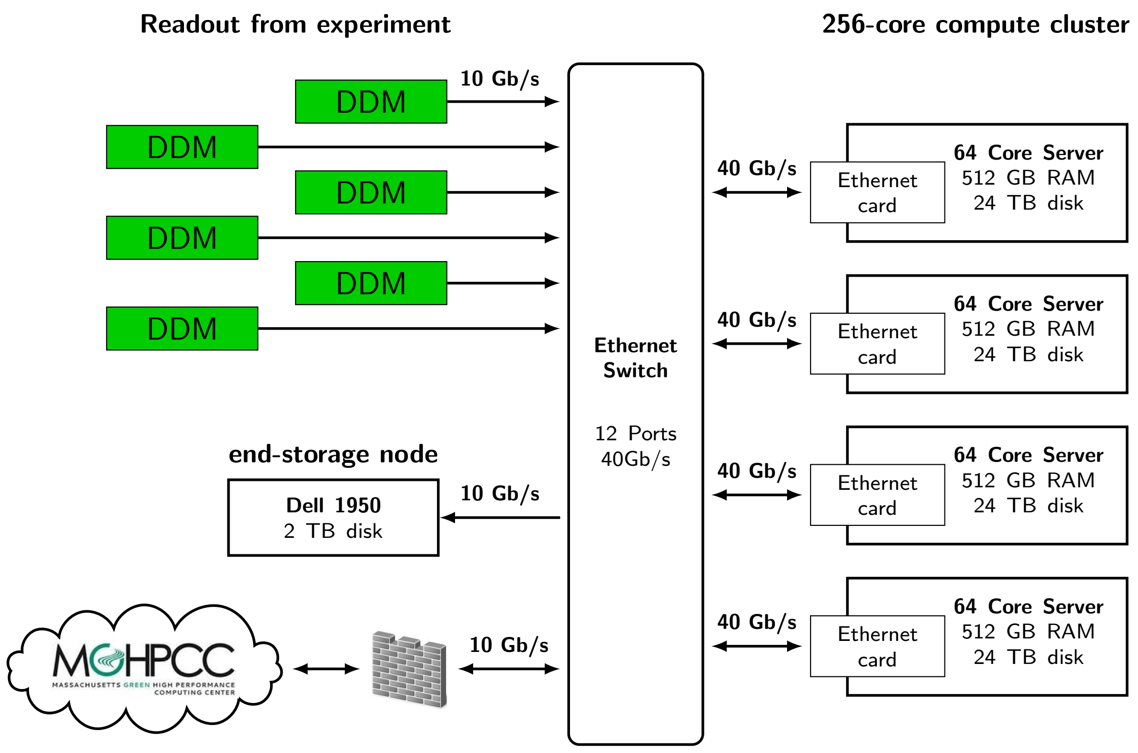

Figure 5: Block diagram of compute cluster for parallel classifying of events. Each compute core on each server receives the data for a common time slice from multiple DDMs. If the load exceeds the computing capacity of that core, events can be either deferred to local disks and/or sent to a remote computer (cloud) for evaluation. To simulate the data end-storage we envisage a small computer.

Typically only a small fraction of the events will be accepted and the raw information from those sent to the end-storage node. We will evaluate different data processing strategies, *e.g.* parallel mode, where each event is analyzed independently by one core, or use data parallelism, where each sub-step of analysis is applied synchronously to multiple events before the next sub-step is performed.[1]

The compute cluster assembled for this project will allow us to deploy and evaluate the timing and the RAM requirements for some of the existing event classification algorithms for comparable-in-size experiments, developed by some of the authors. We would deploy the di-jet finder algorithm implement on the STAR L3-software trigger using a traditional cut-based selection [5], and a variant of a boosted decision-tree trigger algorithm developed for LHCb and intended to be used by the GlueX experiment [6,7]

To improve robustness in the case of the system overload due to either too high input data rate or too slow event processing we will explore two independent fail-safe mechanisms:

- A fraction of events can be temporarily stored on the local disks of the compute nodes for processing at a later time when the data acquisition is halted for other reasons.[2] With the local disk of 24 TB per compute node we will be able to 'defer' some of events for up to 10 hours, assuming only 3/4 the events streamed to the node at a rate of 3 GB/s can be processed instantly.

- Alternatively, one or more virtual machines (VM), similar to the compute nodes with respect to the core count and RAM/core capacity, will be launched at a local private or remote commercial cloud. The fraction of conditionally accepted events can be sent to the those VMs for a time consuming classification and only the (small in size) decision result would be returned. The STAR experiment has exercised the near-real-time data processing on a remote private cloud [8] within the Magellan project. In the actual EIC experiment this "cloud" is likely a common, laboratory-wide computer cluster.

The commodity compute nodes are selected to be on the high-end of today's computing standards: 64 cores per server, 8 GB RAM per core, large local disk, and a 40 Gb/s Ethernet card. The complete list of hardware components required by the compute cluster is listed in Table 1. It is understood, that, for the EIC era, the computing industry will provide next generation CPUs with a much higher number of cores as commodity hardware. The objective is to develop the functioning software

[1] *E.g.*, LHCb model is to process each event on a single core but process 30k events simultaneously.

[2] This approach has been used at the LHCb experiment where it is referred to as a 'deferral' [1].

framework capable to achieve the nominal factory specs of the hardware at a level of 70%. The software is the 'added value' of this segment of this project.

Table 1: Hardware Specification for 256-core compute cluster used in this trigger-less DAQ project.

Equipment	**Qty**
Compute Node 4x 16 cores CPUs, 24 TB local disk w/ writing speed of 500-700 MB/s, 512 GB RAM, single-port QSFP 40 GigE Network Card, example vendor : Microway Navion QuadPuter	4
End-storage + OpenStack Controller Dell 1950, 2TB disk, 1Gb/s Ethernet	1+1
Ethernet Switch 12-ports QSFP 40/56GbE, switching capacity above 1Tb/s, example vendor : Mellanox SwitchX SX1012	1
Network Interface Cards 40 Gigabit Ethernet, example vendor: Mellanox Connect X-3	2 spare
Other Network Cards	
Mellanox 10 Gb/s Ethernet for Dell 1950	2
Cisco Xenpack 10 Gb/s for MIT firewall to the world	1
Copper Cables QSFP-QSFP Passive 40Gb/s, 2m long	4+2 spare
Copper Hybrid Cables QSFP-4xSFP+ passive 40Gb/s, 5m long	1
(with) Mellanox network adapter QSFP-SPF+	1
Rack with shelves, power and cooling	1

3.5 Virtualization and Elastic Cloud Computing

We will add the virtualization layer to compute nodes, preferably using OpenStack[9], to isolate our software from the hardware specific differences and assure reproducibility of event classification locally and on the cloud. The same VM image will run locally and would be transferred and deployed on the cloud facility.

One of the goals of this proposal is to explore the potential issues and overall performance when a fraction of recorded events is classified on remote cloud computing facilities. We will integrate with our virtualized compute cluster and evaluate two such sites:

- MGHPCC at Holyoke, MA[10]
- Commercial Amazon EC2

Running (for brief period at a time) on the Amazon EC2 will allow us to verify the performance of the event classifier algorithm on a 1000+ core cluster, larger than our local 256-core cluster. The Holyoke cluster will let us test how a partnership with a state sponsored local computing cluster works on a longer time scale. We would like to set up a quasi-permanent and cost effective high bandwidth Ethernet connection to the Holyoke computing center and test the long time scale stability of simultaneous local and remote events processing.

The long term goal of exploring the feasibility of elastically scalable cloud computing for the real time data acquisition is to develop a scheme of re-factoring of the offline computing farms at Jefferson Lab or BNL to be partially aggregated with the EIC DAQ. This would allow us to assume a lower safety margin for EIC DAQ peak throughput and hence reduce its cost. Use of virtualization to run the event classification process will enable those explorations.

It is worth noting that LHCb and CMS are using their trigger farms for generating MC when the LHC is down and having this fast connection between the lab-wide and experimental CPUs would enable the trigger farms to effectively become part of the distributed offline Tier-0 site during accelerator down time. The VM-based elastic cloud computing scheme developed within this proposal will allow for 365 days/year use of nominally a "trigger farm". If the VMs work for the trigger running on the central resources then that should work in reverse, i.e., running VMs on the trigger farm during experiment down time can work for the Tier-0 farm for an arbitrary offline tasks.

Some of the authors have experience in setting up and maintaining a local compute cloud, see Fig. 6. We have measured performance on our local OpenStack cluster built on Dell 1950 blades connected via 1 Gb/s Catalyst 4000 switch [11]

- CPU losses due to virtualization of 3.5%, for in-memory computation w/o any I/O to disk
- 4% losses in data transmission from VM to the remote location from the blade hosting this VM vs. from bare-metal OS on the same blade to the same remote location.

The 4% losses in performance expected from virtualization of the compute nodes is more than compensated for by the ease of maintenance and uniform performance that virtual machines provide, especially when the compute nodes may physically reside on remote hardware to which the experiment may have little or no direct access.

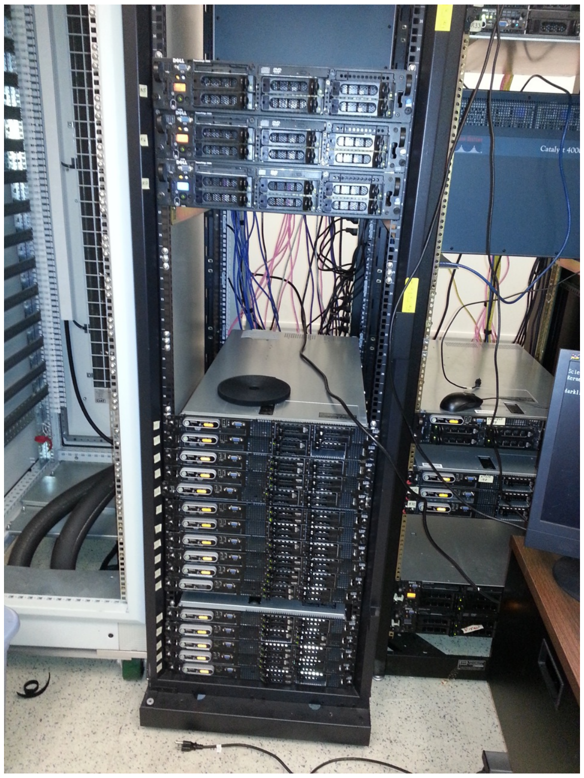

Figure 6: Local OpenStack VM Cluster at LNS/MIT consist of 20+ 8-core Dell 1950 compute nodes, several Dell 2850 controllers, 3 Catalyst 4000 1Gb/s switches. The total size is of 200+ cores and 400+ GB of RAM. Some of those blades and the empty rack seen on the left will be used in the proposed project.

3.6 Miscellaneous Design Objectives

For configuration and control, a (low speed) back channel to the FEE, CM and DDM elements is needed. We want to develop a flexible and scalable protocol for this purpose.

While the average data rate is easy to model, most experiments can produce data in bursts, since the time between events has an exponential distribution. To guarantee no event loss, the bandwidth needs to be calculated with the maximum rate, not the average rate. With buffers and channel bundling, we can reduce this effect and produce essentially deadtime-free operation with a bandwidth closer to the average value. However, it is then possible to have buffer overruns in extreme cases. The design will have accommodation for such an occurrence.

3.7 Scalability

The design is scalable to high channel counts and high event rates, both on a single channel and aggregated over all. The FPGA interconnect speed sets the maximum number of channels on a FEE board. The limit is about 8 GBit/s on a single channel, which would saturate currently available fiber links.

If the data rate of one DDM is exhausted, it is possible to extend the system with multiple, parallel DDMs connected to the same switching fabric. This allows the number of channels to scale almost arbitrarily, limited only by the maximum size of the switching fabric. Since Ethernet is the standard technology used in high bandwidth environments like data centers or internet exchanges, systems with massive backplane bandwidth (multi-TBit/s) and port count are commercially available.

4 Deliverables

4.1 Deliverable Summary

4.1.1 Readout Network

There are three main deliverables for the Readout Network segment of this project. They are:

- Reference firmware for a detector specific front end module
- Reference design and firmware for a data distribution module
- Reference design and firmware for a concatenation module

By producing these reference designs as our main deliverable, we make the project as flexible as possible. Smaller future projects will be able to use the reference designs for their purposes and larger future projects will be able to use the reference designs to inform their own custom designs and thus reduce the engineering needed. This also allows the project to be easily upgradeable as faster and larger FPGAs become available.

4.1.2 Compute Cluster

For the compute cluster the main objective is to develop the software and the scheme for the following action items:

- setup local virtual cluster on the compute nodes
- develop and vet the virtual machine (VM) image running locally, on Amazon EC2, and at Holyoke computing center
- develop and test software running inside VM capable of assembly of the sub-events from DDMs into full events, running sustainably at a high data rate
- implement and vet examples of event-classifier algorithms running on VM
- vet those event classifiers on a 1000+ core cluster
- develop and test the scheme for deferral of event processing if the local cluster is too busy
- develop and test the scheme for sending events to a remote compute cluster for evaluation

4.2 Follow-up Prospects

The compute cluster infrastructure developed for this DAQ proposal has a potential beyond the objectives we are planning to deliver. Those potential tasks also relevant for the future EIC, are listed below, the 'low hanging fruit' which can be selected by new members of this project joining it at a later time and having the man-power resources to work on them.

- Use of compute cluster for Monte Carlo simulations at the time the experiment is not taking data.

- Test of more advanced event classifiers on parallel compute system of 256 cores.
- Experiment on integration of this compute cluster with a subset of machines at Jefferson Lab or BNL computing farms.
- Full design of the front-end electronics for a real detector system.

5 Approximate Schedule

5.1 First year

In the first year we will procure much of the equipment needed to complete the rest of the project. This includes a high speed oscilloscope for board-level debugging, FPGA development boards to be used throughout the development process. Much of this first year will be spent developing a working prototype for the protocol between the different modules in the design and then starting the design of the reference design for the DDM. In parallel, the data generators will be developed for use during the rest of the development.

We will acquire one compute node and the Ethernet switch. This basic setup will allow us to test the FPGA design for the data transfer from the DDM equivalent FPGA development board to the node. In parallel the OpenStack controller deployed on the Dell 1950 (already in our possession) will be connected to the compute node and we will exercise generation, deploying, stability, and agility of the virtual machine image while running processes for receiving data from FPGA. We will exercise instantiation of the same VM image locally and on Amazon EC2.

5.2 Second year

In year two we will procure and test the reference design of the DDM as well as debug and refine the protocol. The CM will be designed, produced, and tested.

Two more compute nodes will be acquired. The compute cluster will then consist of three 64-core blades, and there will be 10 Gb/s connection to the outside world. With three compute nodes one will be able to exercise a non-trivial event building using mock data received from a few DDMs. Also we will establish a high speed data transfer connection with the Holyoke computing center, deploy our VM image there, optimize the data transfer method. In parallel the work on implementation of the STAR event classifier on VMs will proceed.

5.3 Third year

Year three will be used to test out the reference design. This will be thoroughly tested and documented to facilitate technology transfer to other groups interested in using the system. In addition, we plan on investigating how technology not available in year one, but available in year three, can be used in the design.

The compute cluster will reach its full capacity of 256 cores. We will be able to evaluate the full chain of data processing (see Fig. 2) under the full load from all 4 DDMs streaming at 8 Gb/s to 4 compute nodes. We will implement and evaluate performance of the machine learning event filtering. We will demonstrate offloading of the peak computing needs to Holyoke and/or Amazon EC2. We will test the event classifier on a 1000+ core cluster launched on Amazon EC2.

6 Funding Request

To develop this readout system we request funding to cover the design, purchase and assembly of a complete system limited to a relatively small number of simulated detector channels. This will serve to test all aspects of a larger, general purpose readout system.

To perform this work we request support for 0.9 FTE of an electronics engineer spread over three years. The first year will likely require the largest effort in design and procurement. Subsequent years will involve testing, modifying, and fine tuning the design in addition to staying up to date on new developments and technological improvements that could benefit the system being developed.

In addition to the electronic engineer we anticipate the close involvement of post-doctoral physicists and graduate students to take responsibility for programming and testing. To optimize the virtual machine cluster we anticipate the need for a computer science graduate student who would help us for few months per year.

The requested support is outlined in Tables 2 to 4, with estimated costs to cover equipment, component procurement, services, travel and manpower.

N.B. No contingency has been added to these numbers. They represent our best estimate for the actual costs.

Table 2: Equipment, components, and services budget.

Item	2015 (k$)	2016 (k$)	2017 (k$)
Equipment exempt from overhead			
5 GHz digital oscilloscope (LeCroy 760ZI-A series)	70.0		
Reference DDM boards (6)		50.0	
Reference CM boards (6)		40.0	
Compute nodes (4)	18.0	36.0	18.0
Ethernet switch (1)	8.0		
Components subject to overhead*, included			
Dev. boards for DDMs (3, Virtex 7 Series)	21.9		
Dev. boards for Data generation (4, Kintex 7 Series)	3.7	3.7	
Data generation daughter boards (4)		5.5	
Ethernet IP Core licence	5.5		
Development workstation	9.1		
Readout network optical links (10)	1.8	1.8	
Next generation FPGA upgrades			36.6
Additional network cards (2)		2.4	
Copper Ethernet cables (7)	0.4	0.8	0.4
Miscellaneous hardware	2.0	2.0	1.0
Services subject to overhead*, included			
Compute units at Amazon EC2	9.1	9.1	9.1
Existing components in MIT possession			
End-storage node (1)			
OpenStack controller node (1)			
Rack with power and cooling (1)			
Total	149.5	151.3	65.1

* all overheads together are ~81%

Table 3: Manpower and travel budget including overhead.

Item	2015		2016		2017	
	FTE	(k$)	FTE	(k$)	FTE	(k$)
Engineers	0.4	81.5	0.3	63.0	0.2	43.4
Computer science graduate student	0.2	17.1	0.2	17.8	0.2	18.5
Computing conference travel		8.0		8.0		8.0
Total		106.6		88.8		69.9

Table 4: Total budget request including overhead.

Item	2015	2016	2017
Total	256.1 k$	240.1 k$	135.0 k$

7 Summary

With this proposal we plan to develop a modular, scalable, trigger-less, deadtime-free, streaming readout system for data acquisition. The proposal also includes studying the event reconstruction on a multi-core computer cluster connected to the readout system. At the high luminosity planned for the future EIC experiments such a readout and analysis scheme will be needed. It is important to study and specify the scheme now so that the EIC detectors being developed can be designed with the requirements of the readout system in mind.

References

[1] The LHCb Collaboration. LHCb Trigger and Online Upgrade Technical Design Report. *CERN/LHCC 2014-016*, May 2014.

[2] Ananya et al. O^2: A novel combined online and offline computing system for the ALICE Experiment after 2018. *J.Phys.Conf.Ser.*, 513:012037, 2014. doi: 10.1088/1742-6596/513/1/012037.

[3] The ALICE Collaboration. Letter of Intent for the Upgrade of the ALICE Experiment. *CERN/LHCC-2012-012*, September 2012.

[4] A. Accardi *et al.* Electron Ion Collider: The Next QCD Frontier - Understanding the glue that binds us all. *arXiv.org*, December 2012.

[5] Abelev *et al.* Measurement of Transverse Single-Spin Asymmetries for Dijet Production in Proton-Proton Collisions at s=200 GeV. *Phys. Rev. Lett.*, 99: 142003, Oct 2007.

[6] R Aaij, J Albrecht, F Alessio, S Amato, E Aslanides, et al. The LHCb Trigger and its Performance in 2011. *JINST*, 8:P04022, 2013. URL `http://arxiv.org/abs/1211.3055`.

[7] V. Gligorov and M. Williams. Efficient, reliable and fast high-level triggering using a bonsai boosted decision tree . *JINST*, 8:P02013, 2013. URL `http://arxiv.org/abs/1210.6861`.

[8] J. Balewski *et al.* Offloading peak processing to virtual farm by STAR experiment at RHIC. *J. Phys. Conf. Ser.* **368**, *012011 (2012).* URL `http://iopscience.iop.org/1742-6596/368/1/012011`.

[9] OpenStack. Open source software for building private and public clouds. URL `https://www.openstack.org/`.

[10] MGHPCC, Massachusetts Green High-Performance Computing Center at Holyoke, MA [10], dedicated to research computing and funded by the Commonwealth of Massachusetts, Boston University, Harvard University, MIT, Northeastern University, the University of Massachusetts, and private industry. URL `http://www.mghpcc.org`.

[11] J. Balewski. Aggregating local clouds with reusable tools. *Presented at OSG All-hands meeting, April (2014).* URL `http://iopscience.iop.org/1742-6596/368/1/012011`.